EMBEDDED SYSTEM

ELECTRICAL ENGINEERING DEPARTMENT

VEER SURENDRA SAI UNIVERSITY OF TECHNOLOGY

Version 1.0, Dated 17-04-2015

Prepared By

Er B.K.Rana

Assistant Prof., Electrical Engg.
Scheme

Ref 1: Introduction to embedded system: by SHIBU K.V., TMH Publication

Ref 2: Embedded Systems – Architecture, Programming and Design – by RAJKAMAL, TMH Publication

<table>
<thead>
<tr>
<th>Course Item</th>
<th>Ref 1</th>
<th>Ref 2</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. <strong>MODULE _I</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Introduction to Embedded System: What is embedded system</td>
<td>Page 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Embedded systems Vs General Computing system</td>
<td>Page 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3. History of embedded systems, classification of embedded system</td>
<td>Page 5,6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1.3, Sec 1.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4. Major application area of embedded sys</td>
<td>Page 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5. Purpose of embedded system</td>
<td>Page 8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 1.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6. Typical Embedded sys: Core of embedded system</td>
<td>Page 15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chap 2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7. Memory</td>
<td>Page 28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2.2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8. Sensors</td>
<td>Page 35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9. Actuators</td>
<td>Page 35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2.3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10. Communication Interface</td>
<td>Page 45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2.4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11. Embedded Firmware</td>
<td>Page 59</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12. Other system component</td>
<td>Page 60</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2.6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13. PCB and Passive components</td>
<td>Page 64</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sec 2.7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14. <strong>MODULE _II</strong></td>
<td>Page 204</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware Software co-</td>
<td>Chap 7</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
</tbody>
</table>
| 15. | Fundamental Issues in Hardware –Software co design | Page 205  
Sec 7.1 |
| 16. | Comutational model Embedded System | Page 207  
Sec 7.2 |
| 17. | Introduction to unified modeling language (UML) | Page 214  
Sec 7.3  
Page 295  
Sec 6.5 |
| 18. | Hardware software tradeoff | Page 219  
Sec 7.4 |
| 19. | Embedded Hardware design and development : Analog design components | Page 228  
Sec 8.1 |
| 20. | Digital Electronics component | Page 230  
Sec 8.2 |
| 21. | VLSI and Integrated Circuit design | Page 245  
Sec 8.3.1 |
| 22. | Embedded Firmware design and development | Page 302  
Chap 9  
Page 25  
Sec 1.4.8 |
| 23. | Embedded Firmware design approach | Page 303  
Sec 9.1 |
| 24. | Embedded firmware development language | Page 313  
Sec 9.2.2 |
| 25. | Programming in Embedded C | Page 318  
Sec 9.3  
Page 235,237  
Sec 5.1,5.2 |
| 26. | MODULE_III |   |
| 27. | Real Time operating system (RTOS) based embedded system design | Page 381  
Chap 10  
Page 350  
Chap 8 |
| 28. | Operating system basic | Page 382  
Sec 10.1 |
| 29. | Types of operating systems | Page 386  
Sec10.2 |
| 30. | Task process and threads | Page 390  
Sec 10.3 |
| 31. | Multiprocessing and multi tasking | Page 402  
Sec 10.4 |
| 32. | Task scheduling | Page 404  
Sec 10.5 |
| 33. | Thread and process scheduling | Page 422  
Sec 10.6 |
| 34. | Putting task communication | Page 426  
Sec 10.7  
Page 303  
Chap 7 |
| 35. | Task synchronization | Page 442  
Sec 10.8 |
| 36. | Device drivers | Page 476  
Page 2.20 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th>Sec 10.9</th>
<th>Sec 4.9</th>
</tr>
</thead>
<tbody>
<tr>
<td>37.</td>
<td>How to choose RTOS</td>
<td>Page 478</td>
<td>Sec 10.10</td>
</tr>
<tr>
<td>38.</td>
<td>MODULE IV</td>
<td></td>
<td></td>
</tr>
<tr>
<td>39.</td>
<td>Embedded system Development environment (IDE)</td>
<td>Page 556 Chap 13 Sec 13.1</td>
<td></td>
</tr>
<tr>
<td>40.</td>
<td>Types of files generated on cross compilation</td>
<td>Page 588 Sec 13.2</td>
<td></td>
</tr>
<tr>
<td>41.</td>
<td>Dissembler / Decompiler</td>
<td>Page 597 Sec 13.3</td>
<td></td>
</tr>
<tr>
<td>42.</td>
<td>Simulators</td>
<td>Page 598 Sec 13.4</td>
<td></td>
</tr>
<tr>
<td>43.</td>
<td>Emulator</td>
<td>Page 599 Sec 13.4</td>
<td></td>
</tr>
<tr>
<td>44.</td>
<td>Debugging</td>
<td>Page 599 Sec 13.4</td>
<td></td>
</tr>
<tr>
<td>45.</td>
<td>Target Hardware debugging</td>
<td>Page 607 Sec 13.5</td>
<td></td>
</tr>
<tr>
<td>46.</td>
<td>Design Case studies</td>
<td>Page 669 Appendix II</td>
<td>Page 43 Sec 1.10</td>
</tr>
<tr>
<td>47.</td>
<td>Digital Clock</td>
<td>Page 669 Appendix II-1</td>
<td>Page 44 Sec 1.10.3</td>
</tr>
<tr>
<td>48.</td>
<td>Battery operated smart card Reader</td>
<td>Page 699 Appendix II-2</td>
<td></td>
</tr>
<tr>
<td>49.</td>
<td>Automated meter reading system (AMR)</td>
<td>Page 701 Appendix II-3</td>
<td></td>
</tr>
<tr>
<td>50.</td>
<td>Digital Camera</td>
<td>Page 703 Appendix II-4</td>
<td>Page 48 Sec 1.10.4</td>
</tr>
</tbody>
</table>
Assignments

1. Draw IBM PC architecture and describe all components
2. What is difference between Microprocessor and Micro Controller.
3. Describe a CPU which does floating point calculation.
4. Describe Math coprocessor
5. Describe Graphics processor
6. Write 5 general purpose microprocessor and 5 micro controller.
7. Write a page on DSP processor
8. What is PLD and describe commercial types.
9. Write key words of C language. Give 1 line description for each.
10. Differentiate Task, process, thread.
11. Name five commercial operating system and 5 real time operating system.
12. Draw RESET circuit for a CPU.
13. Draw all diagrams used in High level language UML.
14. What is Network Operating System (NOS)? Give it’s seven layer OSI architecture.
15.
16.
17.
Architecture of PC

1. Compatible with 2nd-Generation Intel® Core™ processor family
2. All SATA ports capable of 3 Gb/s, 2 ports capable of 6 Gb/s
Computing System

- Hardware
  - CPU
  - Memory
  - I/O Peripheral
  - Interrupt Controller
  - Timer
  - Display

Operating System
Communication System
Device Driver
Clock Speed

CPU is characterized by address, data, speed.
- RISC / CISC
  - RISC - Reduced Instruction Set Computer
  - CISC - Complex Instruction Set Computer

CPU - Microprocessor
  - Microcontroller

H.W.
What is difference between microprocessor and microcontroller? (2 page)
- CPU operated with floating point calculation
- Math Coprocessor
- Graphics Coprocessor
Commercial CPUs

Intel
- 8051
- 8085
- 8086
- 80186
- 80286
- 80386
- 80486

Motorola
- 68000
- 68010
- 68020
- 68030
- 68040
- Sparc

Glossary of each processor:
- Data Size
- Address Size
- Speed

<table>
<thead>
<tr>
<th>Speed</th>
<th>Data Size</th>
<th>Address Size</th>
<th>Read/Write</th>
</tr>
</thead>
</table>

Memory
- EPROM
- RAM

Data Bus
- 8 bit, 16 bit
- 32 bit, 64 bit

ES-9
What are the Commercial EPROM chip and RAM chip available in the market?

24.07.2014

Hardware

CPU
Memory
I/O device
Interrupt Controller
Timer
Bus Controller

Architecture of General Computing Environment

Development and User Interface Environment

- Operating System
- Communication
- Device Driver

Hardware
Commercial operating systems

Windows
Unix
Linux
Mac
RTOS

**Embedded Environment**

- General, Tonee Characteristics.

RTOS - Real-time operating System.

Soft / Hard,

- S RMS
- Mac - OS - TE
- vx works
- Lynx

<table>
<thead>
<tr>
<th>DIFFERENCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generally Computing System</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Embedded System</th>
</tr>
</thead>
</table>

Section 1.2 Page 4.5 A of system which is combination of a generic hardware and general purpose operating system for executing variety of applications.

A system which is combination of special purpose operating system, hardware and embedded 0s for executing a specific set of applications.
2. Contains general purpose operating system (GPOS)

3. Upgradability and altering program by user

4. Performance is the deciding factor in selection of the system. Always faster memory time.

5. Response requirements are time-critical. Conclusive.

6. Need not be deterministic. So execution behaviour.

2. May or May not have operating system OS as called Embedded OS.

3. For upgrading and altering program you have to go to manufacturer.


5. For certain type of embedded system like mission-critical system. The response time requirement is highly critical.

6. Execution behaviour is deterministic for to certain embedded system called 'Real-time System'.
Classification of Embedded System

4 types of classification:
1. Based on Generation
2. Complexity and performance requirement
3. Based on deterministic behaviour
4. Based on time triggering

Real-time System

\[
\text{Soft} \; \bigwedge \; \text{Hard}
\]

(i) Based on generation:

1. First generation
   - 8 bit microprocessor type
     - 8085
   - 8 or 16 bit microcontroller (Searched packets)
     - 8051
   - No OS required
   - Assembly language programming

2. Second generation
   - 16 bit microprocessor
     - 680x, 681x
   - 8 or 16 bit microcontroller
   - What are 16 bit microcontrollers
Instruction set is complex compared to generation 1.

Embedded operating system

Application

Supervision Control and Data Transmission

DCT & D404 system

3rd Generation:

32 bit microprocessor.
16 bit microcontroller.
DSP - Digital Signal Processing.
ASIC - Application Specific Integrated Circuit.

More complex instruction set, instruction pre-pipelining.

Read instruction and keep in queue

\[ \text{execute instruction} \]

Dedicated embedded real-time operating system like PKE is available.

Intel ReteMone Multitasking operating system.
Application:
- Robotics
- Media
- Industrial process Control
- Communication System.

Fourth Generation:

SOC - System On Chip.

Operating system specific Hardware
and communication specific hardware are integrated with cpu.

Multipole
High performance Real time operating system
Application
- Smart phone devices
- Mobile internet devices
- Industrial process Control
Identification based on complexity and performance

1. Small Scale Embedded System
   - Simple
     - No performance requirement
     - 4 bit, 8 bit, 16 bit
     - Example: Electronic toy application
     - Small

2. Medium Scale Embedded System
   - Low cost, 16 bit, 32 bit
   - Small RTOS performance
   - Industrial controller
   - On Computer Controlled Network
   - Example: Chip = MOTOROLA

3. Large Scale Embedded System
   - 32 bit/64 bit processor
   - High performance is mission critical
   - High performance RTOS
   - RISC
Application of embedded system:

Page 7/8 Section 1.5

2. Household appliances - TV, DVD, Washing Machine, Fridge, Microwave.
3. Home automation and security system - Air conditioner, Alarm, Close Circuit Camera (CCTV).
5. Telecom - Cellular phone and switches.
6. Computer peripherals - Printer, Scanners, Fax machines.
8. Measurement and instrumentation,
   - Digital ORO
   - Logic analysis
   - PLC

9. Health Care
   - Scanner, EEG, ECG

10. Banking and R
    - ATM, currency counter

11. Card Reader
    - Barcode
    - Smart Card Reader

8 01.08.2014

Real world of Embedded System

---

Diagram:

- Input Ports/Sensors
- Memory
- Communication Interface
- Core
- Output Ports (Actuators)
- Other Supporting Integrated Circuit and Subsystems
Core of embedded system

1. General purpose and domain specific processor
2. Microprocessor
3. Microcontroller
4. DSP processors

4. Write a page on DSP processors.

Application specific

5. Integrated Circuits (ASIC)
6. Programmable Logic Device (PLD)
7. Component Commercial off the Shelf Components (COTS)

Write in brief about COTS.

Microprocessor

```
Clock generator

Memory

CPU

Small

Interrupt

Inside

Outside
```
Assignment:

1. Draw Architecture of 8086
2. Draw Architecture of 8051

General Purpose Processor (GPP)
Application Specific Instruction
Set processor (ASIP)

Pentium-

Microcontroller:

It contains

- CPU
- I/O subsystem
- ALU
- CPU
- RAM
- ROM

Examples:

Intel 8089
8088
8051

Home Work:

Difference between microprocessor and microprocessor.
Digital Signal Processing Processors

1. Powerful special purpose 8/16/32/64 bit processors
2. Computational regime or high purpose.
3. 10-150 7-8 times, faster than GPP.

II W, find out a commercial DSP processor.

06.08.2014
RISC and CISC processor

RISC: reduced instruction setComputer.
CISC: complicated instruction set, Computer.

ADD: x + y
Multiply using ADD.

Instead MUL: complicated instruction

30-40 instructions.
Speed is high.
User has power to design complicated instruction.

Page No. 22

Home 82 2 1 1.6
Work difference between RISC and CISC.
Architecture

- Harvard Model
- Von-Neumann model of processor
  - Harvard: code & data separate
  - Von-Neumann: code read only

\[\text{CPU} \quad \text{Code} \quad \text{Data}\]

While executing code, CPU can fetch next instruction and run or pipeline.

Comparisons:

- Harvard architecture
  - Harvard
  - Separate buses for instruction and data fetching

- Von-Neumann architecture
  - Single, shared bus for instruction and data fetching
(3) Faster to populate instruction area faster
(4) Comparatively
     high cost and
cheaper.
(5) No memory
     alignment problem
(6) Since data and
     code memory are
     separate, go
corruption of
     memory possible.

Big-Endian

and Little-Endian processor:

byte 2, byte 1, byte 0.

Memory address

\[ \begin{array}{c|c|c|c}
\text{byte-2} & \text{byte-1} & \text{byte-0} \\
\hline
\text{nl0} & \text{nl1} & \text{nl2} \\
\text{nl1} & \text{nl2} & \text{nl0} \\
\end{array} \]

Little-Endian
Firmware & Tools for Embedded System

Development

See 1.4.2

Table 1.2

Raj Kanal

1. Editors for writing C, C++, Assembly Language Code in computer/PC.
   Allows entry, addition, deletion, insert, appending a previously written code lines or files.
   - Textas C editor
   - Microsoft C editor
   - Windows Environment VS editor. UNIX platform.

2. Interpreters
   for expression - by expression
   or line by line
   translations of written code to machine executable code.

3. Compilers
   Written High-Level Language Code is compiled using Compiler tools and Errors or mistakes are generated in no error than High-level code file.
   Basic converted to machine code file or object file.
4. **Linker**: Application contains more than one file. All files will be compiled. Generated objects file will be linked to form an executable file by **linker**.

5. **Assembler**: For translating assembly mnemonic (or instructions) into an executable file and binary file.

6. **Cross Compiler/Assembler**

   Cross Compiler/Assembler
   Convert source code to binary code using PC/any other computer for different microprocessors.

7. **Simulators**

   To simulate all functions of embedded system circuits including bulk or additional memory and peripherals. It is independent of a particular target system.
8. Source Code Engineering

For source code comprehension, navigation, and browsing:
Editing, debugging & configuring, disassembling and reverse engineering features.
Tools available in the market:
- CVL
- Clearance
- Logoscope

08.08.2014

9. RTOS. Real Time Operating System

Layer 3. Application
Layer 2. API
Layer 1. RTOS
Layer 0. Hardware

This will be taught elaborately in Module 1001.
API: Application Interface

10. The Stethoscope

For dynamically tracking the changes in a programming variable or parameters. It demonstrates the sequence of multiple processes and real-time history.
(11) Trace scope:

To help in tracing the changes in modules and tasks with time in X-axis.

Task

- on
- off

time.

(12) Integrated Development Environment (IDE)

This development software and hardware environment that consists of simulator, editor, compiler assemblers, RTOS debugger, oscilloscope, RTOS emulator, tracer, logic analyzer and application code checker. Burner ofEPROM and Flash.

(3) Prototypes

- Prototype is a small version of the actual large system. This is build to see the behaviour of in less time.

(1) Locator

This uses across assembler output and a memory location map and provide a locator program output as a logfile. It is the final stage of software design of embedded system.
Table 1.3

Application Vs. firmware module requirement.

Module-11
Integrated Circuit Unit

Section-2.5 Embedded firmware

Design Issues of Embedded System

Table 1.8

Page-39
Raj Kamal

1. Power Description
Power consumed in the system.
For many small embedded systems, battery operated systems are used e.g., mobile, digital camera, etc.
Consumption of the battery should be low.

2. Performance

Execution time should be minimized.
Slow memory will be used.
Compiler has optimization option for this purpose.

3 Process Deadlines.

The embedded application will have a number of processes. A process is a piece of code that is scheduled by the kernel. It is created, run, suspended, and terminated based on the scheduling algorithm.

Cost:

![Diagram of process execution]

Process will start with scheduling.

![Diagram of I/O and process]

I/O for process 3, then process 1, and finally process 2.
Interrupt based scheduling. When I/O starts running, the process
process has priority
higher the priority. First of all run.
There are no of process in a system for
example: keypad input processing,
graphic display rewrite.
Audio signal processing and video signal
processing. There are deadlines, within
which each of them may be required
to finish computation and give results.

1. User Interface

2. Keypad
   GUI — Graphical User Interface
   VUI — Visual User Interface

3. Size
   Size of the system is measured in
   terms of
   1) Physical space requirements.
   2) RAM in KB.
   3) Internal flash memory in terms of MB or GB.

4. Engineering Cost:
   Initial cost of developing, debugging,
Set 3 data function.
Set 4 data function.
Set 5 data function.

Any data which is public is prone to corruption.

functions are clustered around data.

Encapsulation.

Data and function together defined into objects.
Class is a template for an object.

```
<table>
<thead>
<tr>
<th>class name</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
</tr>
<tr>
<td>function</td>
</tr>
</tbody>
</table>
```

Objects are programmed by class as object
oriented programming language.

This is a protection.

Encapsulation

Write example of 10 objects and define it in
class of rectangle.

For example - class Geometric shape.

Triangle
Rectangle
S-Circle
function draw
    Calculate area.
    Calculate perimeter.

2) Object
   Can inherit another object.
   Color.
   Drawing pen.
   Drawing pen will include the color.
   That is called inheritance.
Class Person,

Office member.
All office members are also person.
Page 234, Chapter 5, Raj Kamal.
16.10.2014

 Homework

 See 8.3 VLSI and Integrated circuit design
 See 8.7 PCB layout design.

 UML

 Sect 7.3 Introduction

 Object Oriented Design

 Data declaration

 function declaration

 Data and accessing
 function are kept together.
 in a class object is actual.

 Data type are
 Public - usable anywhere
 Protected - used in inheritance.
 private - only external.

 Inheritance is something that permit inclusion
 of one object in another object.
Polygons
  \downarrow \text{Inheritance}
  \hspace{1cm} \text{Rectangle}

- C++, Java are object-oriented programming (OOP)
- Object-Oriented Design (OOD)
  
  UML: Unified Modelling Language.
  
  UML is an OOD
  - Modelled program will be depicted in figure.
  
  WWW.omg.org,
  
  Tutorial of UML: 3.0 or 2.0.
  
  Thinking in C++ (in Internet)

- Homework
  1. Give examples of 10 classes.
  
  2. Discuss 5 feature of embedded C.

Page No - 2750, Table
  Class representation table.

Example: Coffee/Tea Vending Machine.

\[ \text{Coffee} \quad \text{Tea} \]

\[ \text{Sink} \quad \text{Stem} \]
Class: Drink

Start C, T, stop; start, stop;

If (c = 1 & start = 1)

Start

If coffee = on

If start = on

Then vend coffee

If start = on

Stop is pressed

Stop vending

If coffee is on

If & tea = 0 is pressed

Do nothing

If tea = 0

If start = 0

Then vend tea

If start = 0

Stop is pressed

Stop vending

If coffee, tea is on

If coffee is pressed

Do nothing
UML: State transition Diagram

Program has N number of states, M number of events.

Based on the event, the program will transit from current state to next state.

Möller model

Moore model.

Switch (state) (event)

\[
\begin{align*}
\text{case of } S_1: & \quad \text{switch (event)} \\
& \quad \text{case of} \\
& \quad c_1: \quad \text{call } f_1() \\
& \quad c_2: \quad \text{call } f_2() \\
& \quad c_3: \quad \text{call } f_3() \\
\end{align*}
\]

Study fig. 7.5 page 109,

FSM for a Coffee and Tea vending machine
Machine has 4 states:

- State A: start for coin
- State B: coin inserted
- State C: dispense tea
- State D: dispense coffee

Event: TEA dispensed
- Action on VI
- Event: Cancel
- Action on C
- Event: Cancel
- Action on C

Coin operated telephone system: page 211, Fig. 26

FSM of coin operated

State transition table:
Ed State transition table for previous, next.

<table>
<thead>
<tr>
<th>State</th>
<th>$s_1$</th>
<th>$s_2$</th>
<th>$s_3$</th>
<th>$s_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e_3$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e_4$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e_5$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$e_6$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Message Sequence Diagram (MSD)

Two agents can communicate with each other by two ways: message which are sequenced.

![Message Sequence Diagram](image)

Railway ticket bus reservation system.

![Railway Ticket Bus Reservation System](image)
Data flow diagram (DFD)

Process will receive and send data.

Data will flow from one process to another process onto system.

Send data
Receive data
Process data
and take action.

Collaboration diagram

P1, P2 may be on one complete or different complete.

For example - client server
Resources

Server

Client 1

Client 2

Client 3

Read chapter 6, page 873 (Rajkumar).
Chapter 9  Embedded Firmware Design and Development

Two approaches:
1. Super loop model
2. Operating system (OS) based approach.

Process/Task

- Execution of process
- Time
- Round robin
- Scheduling
- Numbers (0, 1, 2, 3)
- Preemptive scheduling
- When high priority task/process is ready
- Low priority task will be stopped, high priority
- Task will resume
- In super loop, there is no OS
- Linear coding of the program is done.

Task 1
Task 2
Task 3
Task 4

ES-41
Operating System based

Real-time OS

HARDWARE

Task management
Time management
Interrupt management

Module 1-11

Real-Time operating System

Functions of operating systems are:
1. Process/Task management
2. Memory management
3. Time management
4. File management
5. I/O subsystem application

One cannot access kernel layer without API. This layer is protected.
Large systems service
Signal service - MICRO KERNEL.
OS

<table>
<thead>
<tr>
<th>General Purpose OS</th>
<th>RTOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(OS)</td>
<td>(Real-time OS)</td>
</tr>
</tbody>
</table>

Ref:
Chapter 10

Process/Task Management

Processes/Task are code piece running concurrently in application and scheduled by kernel.

$p_1, p_2, p_n$

Process:

- Process start
- Process
- Process end

States of process:
- Created
- Ready to run
- Running
- Completed
- Blocked
State transition diagram of process

Fig 10.6

State Transition diagram

(created)

Intercepted to memory.

Ready

Interrupted on preempted, scheduled for execution.

Blocked

Waiting

for I/O.

Runnable

Execution completed.

Each process or task has a process/task control block (PCB/TCB).

It is a piece of data used in management.

<table>
<thead>
<tr>
<th>Task ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Name</td>
</tr>
<tr>
<td>Priority</td>
</tr>
<tr>
<td>Start of Code</td>
</tr>
<tr>
<td>End of Code</td>
</tr>
<tr>
<td>Stack Pointer</td>
</tr>
</tbody>
</table>
Proces and Task Management:

$symb/Tcb$

create task (task_id, TCB address, name priority, code address)

Code address:

task_id — integer
TCB address — memory pointer
name — string
priority — integer (0-15)

code address — memory pointer to code

run — task (task_id)
It will put the task in ready to run state

suspend — task (task_id)
It will put the task in dead, suspended or blocked state.
delete task (task_id)
This will delete task from memory.

Task function
{
    while (1)
    if task...
        ...
    
    Thread...
    If a small task subjected to round robin scheduling:
create thread
run thread
stop thread
Context-look (G) — Task scheduling is blocked.
Context-unlock (G) — Task scheduling.

Memory Management

```
Global Space:

Total:

Free memory:

Used:

Memory management:

Task:

Virtual memory = 2 GB
```

Garbage Management

```
MIB memory task
Start/End from task
```

Create memory
Delete memory
Free memory
Timer management

```
8255
Timer
Clock

Create-timer (timer-id, time-value)
Stop-timer()
Run-timer()
Sleep() -> Sleep definite
Sleep indefinite
```

Code

Sleep()

Interrupt management

```
Install-ISR (ISR address, N)
Delete-ISR (Dummy code/ address N)
```

Device Management

29/10/2011

```
I/O Subsystem/ Device Management

4 generalized I/O device
Real work + configure

ISR
Device

We have to do device management from operating
System like PCB, we need a data block for device
called PCB.
```
<table>
<thead>
<tr>
<th>Id</th>
<th>open-device()</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>close-device()</td>
</tr>
<tr>
<td>Address</td>
<td>enable-device()</td>
</tr>
<tr>
<td>ENABLED/DISABLED</td>
<td>delete-device()</td>
</tr>
<tr>
<td>Status</td>
<td>control-device()</td>
</tr>
<tr>
<td>statistics</td>
<td>read-compute</td>
</tr>
<tr>
<td></td>
<td>write</td>
</tr>
</tbody>
</table>

Initialize
Control
Configure
Send-command
Get-status

Given a device
This cell should be
coded to use in os.
and applications
known as device driver

Examples

Printer driver on pc
This is loaded and used.

8255 The device 8255

Command
IN
Mode 0 ?
Mode 1 ?
Mode 2 ?
Enter process Communication

In applications where multiple processes and threads:

- Process 1
- Thread 1
- Thread 2
- Process 2
- Thread 1
- Thread 2
- Process N
- Thread 2
- Thread 3

One should have a mechanism to talk between the process.

One should have a mechanism to share data called PDB:

**Data sharing**

- Global data shared
- semaphore
- Counting semaphore

**Semaphore**

- **semaphore
  * T/P = 0.1**

**Semaphore commands**

- **get - Global address, (name, address)**
- **create - semaphore**
  - **get - semaphore**
  - **delete - semaphore**

**In counting semaphore, value of semaphore (0-N)**

**Process Communication**

- (1) Mailbox
- (2) Pipe
- (3) Socket
Message Queue

Interprocess Communication

- BD
- top
- bottom

next message

no of messages

Messages

message

One can put a

1. Message queue
2. Mailbox
3. Pipe
4. Socket
5. RPC

One can put a message in the linked list.
One pick a message from linked list.

Ref: Hitl Rajkanal.

Create - Q
Post - Q
Accept - Q

peek - Q
push - Q

top - Q
A queue is a FIFO (first in, first out) data structure. It will wait on a to read the message.
flush/qs deletes all messages and statistics on the qs.

Top-Q: Return the pointer of top of the qs of particular message present in the qs or deleted.

Query - Q
A coparticular message is deleted.

Mailbox

create-mailbox
post - mailbox
accept - mailbox
print - mailbox
query - mailbox
Typically used when large
suitable for large data
file data.

Pipe: create, open, write, read, connect, error.

Socket

Communication layer

Server

Client

Ref. 7:14 RajKumal

Port ☻ Port A

ES-PA